Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT®
Discrete Applied Mathematics 137 (2004) 127–153

DISCRETE
APPLIED
MATHEMATICS

www.elsevier.com/locate/dam

Gossiping and broadcasting versus computing functions in networks☆

Martin Dietzfelbinger

Technische Universität Ilmenau, D-98684 Ilmenau, Germany

Received 3 October 1999; received in revised form 19 June 2002; accepted 19 October 2002

Abstract

In the theory of dissemination of information in interconnection networks (gossiping and broadcasting) one assumes that a message consists of a set of distinguishable, atomic pieces of information, and that one communication pattern is used for solving a task. In this paper, a close connection is established between this theory and a situation in which functions are computed in synchronous networks without restrictions on the type of message used and with possibly different communication patterns for different inputs. The following restriction on the way processors communicate turns out to be essential:

(∗) "*Predictable reception*": At the beginning of a step a processor knows whether it is to receive a message across one of its links or not. We show that if (∗) holds then computing an n-ary function with a "critical input" (e.g., the OR of n bits) and distributing the result to all processors on an n-processor network G takes exactly as long as performing gossiping in G. Further we study the complexity of broadcasting one bit in a synchronous network, assuming that in one step a processor can send only one message, but without assuming (∗), and broadcasting one bit on parallel random-access machines (PRAMs) and distributed memory machines (DMMs) with the ARBITRARY access resolution rule.

Keywords: Communication in networks; Dissemination of information; Lower bounds; Synchronisation; Parallel random access machines

☆ The results of this paper have appeared in preliminary form in the proceedings of STACS'97, Springer Lecture Notes in Computer Science, Vol. 1200, Springer, Berlin, pp. 189–200.

E-mail address: martin.dietzfelbinger@tu-ilmenau.de (M. Dietzfelbinger).

0166-218X/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0166-218X(03)00257-9

1. Introduction

The purpose of this paper is to demonstrate that the well-established theory of gossiping and broadcasting in interconnection networks can directly be applied to obtain lower bounds for algorithms that compute functions in synchronous networks consisting of processors connected by bidirectional links. In this introductory section, we informally discuss the relevant structures and describe the results. Formal definitions will be given in Section 2.

1.1. Gossiping and broadcasting

In *gossiping* theory one deals with the following basic situation: each node of a network initially has an atomic piece of information; the purpose of a gossiping scheme is to distribute all the information to all the nodes. For this, in rounds, the nodes send each other messages consisting of an arbitrary number of pieces of information. The standard restriction is that in one round a node can communicate with only one of its neighbors. One distinguishes 1-way (or half-duplex) mode, where in a round information can be sent through a link in only one direction, and 2-way (or full-duplex) mode, where in a round two nodes may exchange all their information through a link that connects them. The most intensively studied efficiency criterion in this theory is the number of rounds needed for disseminating all pieces of information to every node. The *broadcasting* problem is similar, excepting that only one piece of information, initially located at one node, is to be spread to all others. The *accumulation* problem is the converse of the broadcast problem: the aim is to collect all pieces of information initially located at the single nodes in one distinguished node.

Since the actual contents of the ideal "pieces of information" are irrelevant for the task, the algorithms considered are communication schemes that do not depend on any input data.

For an account of the history of the area of gossiping and broadcasting and the intensive research devoted to it see the survey [16], the more recent surveys [13,19], or other articles in the special issue of *Discrete Applied Mathematics* (vol. 53, 1994). In the fundamental paper [22], which also contains many lower bound arguments, the relevance of the gossip model for real multiprocessor systems is discussed.

1.2. Computing functions in processor networks

In this paper we consider the problem of computing functions in networks. The computational model is a network of n processors, $P_1,\ldots,P_n$, that are connected by bidirectional links, according to a network graph $G=(V,E)$. The processor network is to compute an n-ary function $f:A_1\times\cdots\times A_n\to A$, for arbitrary sets $A_1,\ldots,A_n$, and A, in the following way. Initially, processor P_i knows the ith component a_i of the input $a=(a_1,\ldots,a_n)$; at the end, all processors know the result $f(a)$ ("global output"). (For example, for realizing a synchronization barrier in the network, i.e., in order to find out whether all processors have finished certain subtasks from a set $\mathscr{T}_1$ given to them and inform them whether to proceed to another set $\mathscr{T}_2$ of subtasks that requires that

all activities for $\mathscr{T}_1$ are finished, the Boolean function OR must be computed with global output.) We will also consider the situation where only one processor has to know the result ("local output"). The processors work synchronously in lock-step, i.e., in global steps $t = 1, \ldots, T$. In one step, a processor may communicate with at most one of its neighbors. Different models are obtained by allowing only 1-way traffic on a link in a step (half-duplex mode) or 2-way-traffic (full-duplex-mode).

Fixed versus data dependent communication. A gossiping scheme consists of a single, fixed communication pattern, whereas an algorithm that computes a function may use different communication patterns for different inputs. It is almost obvious that a gossiping scheme can be used to compute any function with global output (cf. Observation 3.1). In [22] it is stated that lower bounds for gossiping carry over directly to the synchronization problem or, even more generally, to computing any multiple-output function in which all output components depend on all input components, like matrix inversion or computing the discrete Fourier transform of a vector, on arbitrary processor networks. However, although the task of computing a function that depends on all input components with global output bears some resemblance to the gossiping problem, there does not seem to be an obvious way of identifying a gossiping scheme within an arbitrary algorithm for such a function.

To illustrate this, consider the function

$$h : \{0,1\}^n \to \{0,1\}, \quad (a_1, \ldots, a_n) \mapsto \bigvee_{1 \leqslant i \leqslant m} \bigwedge_{1 \leqslant j \leqslant m} a_{(i-1)m+j},$$

for some square number $n = m^2$. If $h(a_1, \ldots, a_n) = 1$, it is sufficient to tell all nodes that $a_{(i-1)m+1} = \cdots = a_{im} = 1$ for *some* i; if $h(a_1, \ldots, a_n) = 0$, it is sufficient to inform all nodes that $a_{1,s(1)} = \cdots = a_{m,s(m)} = 0$, for suitable $s(1), \ldots, s(m) \in \{1, \ldots, m\}$. It is conceivable that there are network algorithms that employ different communication patterns for different inputs to compute this function in fewer steps than the number of steps in a gossiping scheme.

In this paper we give precise sufficient conditions for when one can indeed find a gossiping scheme within a network algorithm, and thus provide a formal version of the statement from [22] mentioned above.

There is an essential restriction on the communication mode in the network, which turns out to be necessary for our results to be true; it can be phrased informally as follows.

(∗) "*Predictable reception*": a processor knows at the beginning of a step whether it is to receive a message across one of its links in this step or not.

That means, a processor must not, in step t, wait for a message that may or may not arrive and proceed in different ways when a message arrives and otherwise. This restriction makes it impossible that information is transferred by *not* sending a message in a certain round. The relevance of such a possibility in the context of computations on the parallel random-access machine (PRAM) was observed in [7], and investigated in more depth in [3,9]. Apart from this restriction, the model is quite general, e.g., a

processor may wait for a message from any one of its neighbors without specifying the sender or may make a message available to any neighboring processor that wants to receive it. As will be discussed later, algorithms that observe restriction (∗) are suitable to be executed on asynchronous networks as well (Remark 2.9).

1.3. Results

It is almost obvious that if a network has a T-round gossiping scheme then it can solve the synchronization problem in T steps with 1-bit messages and trivial computation, and, under the assumption that messages may be arbitrarily long and computation is for free, it can compute any function with global output in T steps. The main result of this paper essentially is that this is best possible. More specifically, we show the following for processor networks with "predictable reception".

- If an n-ary function f that has a *critical input* a, i.e., an input with the property that each of the n components of the input can be changed in such a way that the value of the function changes, can be computed on a processor network G with global output in T steps, then G has a T-round gossiping scheme.
- If an n-ary function f that has a critical input can be computed on a processor network G in T steps with local output at processor P_{i_0}, or a function f that depends on component a_{i_0} can be computed on G in T steps with global output, then G has a T-round broadcasting scheme.

These results hold if both in the processor network and in the gossiping network 1-way resp. 2-way communication is assumed. Of course, the idea of the proof is to show that any communication pattern that the network produces on the critical input in fact essentially is a gossiping scheme respectively a broadcasting scheme. Although this is quite intuitive, the result does not seem to be obvious, due to the fact that communication patterns for different inputs may be different. The role played by the restriction "predictable reception" in clearing away these different communication patterns becomes apparent only in the proof.

In Sections 4 and 5 we will consider networks without restriction (∗). As for computing functions with global or local output, we shall see that the possibility of using different communication patterns on different inputs actually makes it possible to compute the OR faster than by a gossiping respectively an accumulation scheme. Still, it can be shown that in no network the speedup can be more than a factor of 4.

In Section 5, we will exactly characterize the complexity of broadcasting a bit in a synchronous network with very general communication rules in terms of a variant of broadcasting schemes in the gossiping type model. The only remaining restriction is that in one step a processor may send only one message that may not be duplicated by the communication system.

1.4. Applications

The consequences of our results for networks with "predictable reception" are threefold. First, as corollaries we obtain a host of lower-bound results for computing

functions in networks of different topologies with algorithms that obey restriction (∗), since all lower bounds proved for gossiping and broadcasting carry over. As is common in gossiping theory, many of these bounds are tight. E.g., we obtain the following:

- Computing the OR in 1-way mode with global output on a complete network of n processors takes $1.44\ldots\log n \pm O(1)$ steps [1] [11,22,23,28].
- Computing the OR in 1-way mode with global output in a ring of n processors takes time $n/2 + \sqrt{2n} \pm O(1)$ [18,22].
- Computing the OR in 1-way or 2-way mode with global or local output on a butterfly network with $n = k \cdot 2^k$ nodes takes at least $1.7417k$ steps [21].
- Computing the OR in a complete k-ary tree of depth d with local output at the root takes exactly kd steps; with global output, $2kd$ steps are necessary and sufficient [19].

Second, it follows that the so-called minimum broadcast subgraphs (cf. [16]) of a network or the trees and schedules used to achieve optimal broadcast times are also the optimal communication pattern for computing functions with a critical input on the corresponding networks, with local output, under restriction (∗). The analogous statement for minimum gossip graphs (cf. [16]) also holds. Third, it has been an objection to upper bounds described in gossiping theory that the model used is somewhat unrealistic in that it allows to send messages containing an arbitrary number of atomic pieces of information from one node to another in one round. Thus, claims to the effect that a lower bound was optimal by giving the matching upper bound were not completely satisfying. (See [1] for an investigation of gossiping with bounded packet sizes.) In our network model, the corresponding upper and lower bounds hold for the problem of computing the OR function, which can be solved by sending messages consisting of one bit. Thus, both upper and lower bounds hold for a quite realistic model.

The model without "predictable reception" is general enough to make it possible to draw conclusions about the complexity of broadcasting a bit in some other models, viz., PRAMs and distributed memory machines (DMMs), with communication that obeys the EXCLUSIVE READ rule. Not surprisingly, the mode of writing is not relevant in this context. (See [12,20] for information on the complexity of PRAM computations, and [10] for a specification of the DMM model with various access conflict resolution rules.)

1.5. Related work

In [8], the relevance of results for the gossiping problem for real multiprocessor systems was discussed in depth, but informally, i.e., without making the model for the multiprocessor system explicit. To the best of the knowledge of the author, the problem considered here has not been studied before on a comparable technical level, with the exception of work done by Belting in his diploma thesis [4], who used a

[1] All logarithms are to the base 2, unless noted otherwise.

combination of the lower-bound method for the gossip problem on the complete network from [11,22,23,28] with the degree technique for proving lower bounds for CREW PRAMs from [8] to show that computing the OR function in networks that obey restriction (∗) and the exclusive-write property (it is forbidden that in one step, more than one message is sent to one processor) takes exactly as long as gossiping in this network. The proof method used in the present paper is completely different. Our technique is, however, related to the method introduced in [31] for analyzing computations on concurrent-read concurrent-write PRAMs with bounded communication width. The method for analyzing broadcasting algorithms in networks with unrestricted communication is a new and more general formalization of the idea of analyzing PRAM computations by keeping track of those cells and processors that are "affected" by some input bit, which has been used in [3,7].

In [2], the problem of computing and distributing the value of a commutative and associative function in a complete network is considered, with the variant of communication mode that allows a processor to send one message *and* receive one message in one step. The lower bound $\lceil \log n \rceil$ claimed in that paper for the broadcasting problem is called "obvious", and it seems that a fixed communication pattern is assumed. Similarly, the lower bound mentioned in [5] for the more general "k-port" model seems to be based on this assumption.

Also, it should be mentioned that in the context of *asynchronous* communication in networks, Tel studied a family of algorithms that were all recognized to be equivalent (called "normal algorithms" in [29] and "wave algorithms" in [30]). They would correspond to the problem of computing functions that have a critical input with "local output". However, because of the absence of a notion of time in these models, Tel's results do not have direct applications in our setting.

1.6. Structure of paper

The structure of the rest of this paper is as follows. In Section 2, we recall the relevant notions from gossiping theory and formally describe the network model. Section 3 contains the proof of the main theorem; Section 4 contains variants of the main result, in particular, extensions to functions that do not have a critical input, to multiple-valued functions, and to networks without restriction (∗). Finally, in Section 5 we consider the problem of broadcasting one bit in unrestricted networks, and mention applications to other parallel computational models (exclusive-read PRAMs, and distributed memory machines (DMMs) with the ARBITRARY read and write conflict resolution rule).

2. Preliminaries

In this section, we first recall some basic definitions and facts from gossiping and broadcasting theory. Next, we give a precise description of the type of processor networks we will consider. For completeness, we also recall the definition of some well-known complexity measures for n-ary functions.

2.1. A formal view of gossiping and broadcasting

For the rigorous definitions of gossiping, broadcasting, and accumulation schemes we partly follow the survey paper [19]. (See that paper as well as [13] for more information on the subject, in particular for lower and upper bounds for specific networks, and for variations of the models.) Throughout this paper, by a graph $G = (V, E)$ we mean an undirected graph without loops or multiple edges, where, without loss of generality, $V = \{1, \ldots, n\}$ for some $n \in \mathbf{N}$. An undirected edge between nodes i and j is denoted by $\{i, j\}$, a directed edge from i to j by (i, j).

Let $G = (V, E)$ be such a graph, and assume each node $j \in V$ has a piece π_j of information. A communication scheme works in rounds $1 \leqslant t \leqslant T$, where in each round a node may send (copies of) all pieces of information it has collected so far to one of its neighbors. In gossiping, after round T every node must know all pieces of information. Such a communication scheme may be compactly represented by fixing for each round t the set M_t of the edges along which messages are sent. For most of this paper, we only allow schemes in which in one round a node can communicate with only one of its neighbors. Quite naturally then, the edge sets M_t are described as (*undirected* or *directed*) *matchings* in G, according as the communication along the edges is in 1-way or in 2-way mode.

Definition 2.1. Let $G = (V, E)$ be a graph.

(a) An *undirected matching* in G is a set $M \subseteq E$ consisting of node-disjoint edges.
(b) A *directed matching* in G is a set M of node-disjoint edges, where each edge is given a direction.
(Formally,
(i) $M \subseteq \{(i, j) | \{i, j\} \in E\}$;
(ii) if (i, j) and (i', j') are distinct elements of M then $\{i, j\} \cap \{i', j'\} = \emptyset$.)

Definition 2.2. A *communication protocol* for a graph $G = (V, E)$ in 1-way [resp. 2-way] mode is a sequence $\mathcal{M} = (M_1, \ldots, M_T)$ of directed [resp. undirected] matchings of G. The number T is called the number of rounds of protocol $\mathcal{M}$.

Node i will send all its information to node j in round t if $(i, j) \in M_t$ (1-way mode) resp. $\{i, j\} \in M_t$ (2-way mode). For later use, we need notation for the pieces of information that the nodes have collected after t rounds of the communication protocol have taken place. If we intend $K_t(i)$ to be the set of those indices $j \in V$ such that node i knows π_j by the end of round t, the following inductive definition is the straightforward formalization.

Definition 2.3. Let $\mathcal{M}$ be as in Definition 2.2. The sequence $\mathcal{K}(\mathcal{M}) = (K_0, K_1, \ldots, K_T)$ of mappings $K_t : V \to \mathcal{P}(V)$, where $\mathcal{P}(V)$ denotes the power set of V, is defined as follows:

(i) $K_0(i) = \{i\}$, for $i \in V$,

(ii) For $1 \leqslant t \leqslant T$, $i \in V$:

$$K_t(i) = \begin{cases} K_{t-1}(l) \cup K_{t-1}(i) & \text{for the (unique) } l \in V \text{ such that} \\ & (l,i) \in M_t \quad [\text{resp. } \{l,i\} \in M_t], \\ & \text{if such an } l \text{ exists.} \\ K_{t-1}(i), & \text{otherwise.} \end{cases}$$

Definition 2.4. Let G be a graph, $\mathscr{M}$ a communication protocol for G in 1-way [2-way] mode. Let $\mathscr{K}(\mathscr{M})$ be the sequence associated with $\mathscr{M}$ as in the previous definition.

(a) $\mathscr{M}$ is a *gossip protocol* for G in 1-way [2-way] mode if $K_T(i) = V$ for all $i \in V$;
(b) $\mathscr{M}$ is a *broadcast protocol* for G in 1-way mode with source node $i_0 \in V$ if $i_0 \in K_T(i)$ for all $i \in V$; and
(c) $\mathscr{M}$ is an *accumulation protocol* for G in 1-way mode with target node $i_0 \in V$ if $K_T(i_0) = V$.

Definition 2.5. Let $G = (V,E)$ be a graph:

(a) The 1-way [2-way] *gossip complexity* $r(G)$ $[r_2(G)]$ of G is the minimum T such that there is a gossip protocol for G in 1-way [2-way] mode with T rounds.
(b) The (1-way) *broadcast complexity* $b(G,i_0)$ and the (1-way) accumulation complexity $a(G,i_0)$ are defined analogously.

One can also define broadcast and accumulation complexity in 2-way mode. However, it is easily seen that these do not differ from their 1-way counterparts. In fact, we even have the following. (For the simple proofs of these claims, see, e.g., [19].)

Fact 2.6. *$a(G,i_0) = b(G,i_0)$ for all $G = (V,E)$ and $i_0 \in V$.*

2.2. Specification of the network model

In this section, we describe the machine model that will be the basis of our considerations. The description will be slightly informal; a fully rigorous definition (involving abstract state sets, transition functions, read-address, write-address, and write-value functions, as well as input and output functions) can quite easily be constructed, e.g., along the lines of the formal description of a CREW PRAM in [7,9]. We consider a network consisting of n processors, $P_1,\ldots,P_n$, and of a set of bidirectional links, each of which joins two of the processors. The topology of the network is described by a graph $G = (V,E)$ with $V = \{1,\ldots,n\}$, the edges $\{i,j\} \in E$ representing the links. Assume the network is to compute a function $f : A_1 \times \cdots \times A_n \to A$. At the beginning of the computation, input $a_i \in A_i$ is given to processor P_i, for $i \in V$, that means, the initial state of P_i depends on a_i.

The computation proceeds synchronously in steps $t = 1,\ldots,T$. The actions of the network in one step are as follows. We first describe the 1-way case (half-duplex use

of links). Processor P_i, $i \in V$, on the basis of its state after step $t-1$, chooses one of the following two possibilities.

(S) Choose a message $m_{i,t}$ and a set $V_{i,t} \subseteq \{j \in V \mid \{i,j\} \in E\}$ representing possible recipients of the message. We say that P_i **SENDS** a message in this step. (For notational convenience, we also allow that $V_{i,t} = \emptyset$, which means that P_i does nothing in this step.)

(R) Choose a nonempty set $W_{i,t} \subseteq \{l \in V \mid \{l,i\} \in E\}$ of neighbors, representing possible senders from which P_i wishes to **RECEIVE** a message.

Now, messages are transmitted, so as to satisfy the choices made by the processors, as described in the following. Consider the set

$$E_t = \{(i,j) \mid j \in V_{i,t} \text{ and } i \in W_{j,t}\},$$

representing all edges across which information may flow. It will be important that we may choose from E_t disjoint edges to be used for communication in a greedy manner, and still are guaranteed that all processors that want to receive a message actually get one. For this, condition $(*)$ is formulated technically as follows:

$(*)^1$ "*Predictable reception* (1-*way*)": For any directed matching $M' \subseteq E_t$ there is a directed matching M with $M' \subseteq M \subseteq E_t$ that covers all recipients, i.e., if P_j has chosen to receive a message in step t then $(i,j) \in M$ for some $i \in V$.

Some matching $M \subseteq E_t$ that covers all recipients is chosen (by "the system"), and for each pair $(i,j) \in M$ message $m_{i,t}$ is delivered to P_j. Messages that are not delivered are discarded. We require that no matter which decision is made here by "the system", the output produced at the end of the computation is always correct. (This rule is analogous to the ARBITRARY write-conflict resolution rule for PRAMs, cf. [20].) Processors P_i that send a message in step t change their state only by noting that step t is finished; those processors P_j that have received a message $m_{i,t}$ assume a new state that also depends on this message. After the last step, T, the result $f(a)$ is known to all processors (in "global output" mode) or to one designated processor P_{i_0} (in "local output" mode), that means, $f(a)$ is a function of the state that is finally reached by each processor resp. by processor P_{i_0}.

Remark 2.7. We sketch two special combinations of possible choices for the sets $V_{i,t}$ and $W_{j,t}$ that may make it clearer which variety of possibilities is covered by this model.

(a) All senders P_i might specify a set $V_{i,t}$ with $|V_{i,t}| = 1$ or $V_{i,t} = \emptyset$, and all recipients P_j might specify $W_{j,t} = \{i \mid \{i,j\} \in E\}$: this corresponds to the situation where each receiving processor has a "write window" into which other processors may write; conflicts are resolved by the ARBITRARY rule known from PRAMs. Requirement $(*)^1$ simplifies to the condition that for each recipient P_j there must be at least one message actually addressed to it.

(b) All recipients P_j might specify a set $W_{j,t}$ with $|W_{j,t}| = 1$, i.e., they are interested in seeing a message from a specific neighbor, and all senders P_i might specify $V_{i,t} =$

$\{j \mid \{i,j\} \in E\}$, i.e., make their respective message available to all their neighbors: this corresponds to a situation in which senders offer their information via a "read window" accessible to all their neighbors. Requirement $(*)^1$ turns into the EXCLUSIVE READ rule known from PRAMs.

In the 2-way (full-duplex) variant of the model, the basic structure of the computation is similar. However, here at the beginning of step t processor P_i fixes a message $m_{i,t}$ and a set $V_{i,t} \subseteq \{j \in V \mid \{i,j\} \in E\}$ of neighbors that are possible partners. (If $V_{i,t} = \emptyset$, the processor does not want to communicate.) Consider the graph $G_t = (V_t, E_t)$, where $V_t = \{i \mid V_{i,t} \neq \emptyset\}$ and $E_t = \{\{i,j\} \mid i,j \in V_t \text{ and } i \in V_{j,t} \text{ and } j \in V_{i,t}\}$. The condition "predictable reception" here takes on the following form.

$(*)^2$ "*Predictable reception* (2-*way*)": Any partial matching $M' \subseteq E_t$ can be extended to a perfect matching M for G_t, i.e., to a matching in G_t that covers all nodes in V_t and includes M'.

The "system" (arbitrarily) chooses one such perfect matching M, and the processors communicate according to this matching, i.e., for every edge $\{i,j\} \in M$ message $m_{i,t}$ is delivered to P_j and message $m_{j,t}$ is delivered to P_i. Processors that have received a message change their state accordingly.

Definition 2.8. Let $G = (V,E)$ be a graph specifying a processor network, and let f be an n-ary function.

(a) A network algorithm (in 1-way or 2-way mode) is said to compute f with "*global output*" ("g") if for all inputs a, after T steps have been performed, all processors know $f(a)$.
(b) The 1-way network complexity $T_G^{1,\mathrm{g}}(f)$ is the minimum number T of steps of a 1-way algorithm with global output that computes f. The 2-way complexity $T_G^{2,\mathrm{g}}(f)$ is defined analogously.
(c) A network algorithm in 1-way or 2-way mode is said to compute f with "*local output*" ("l") at node P_{i_0} if after step T of the algorithm processor P_{i_0} knows the result $f(a)$.
(d) The network complexities $T_{G,i_0}^{1,\mathrm{l}}(f)$ and $T_{G,i_0}^{2,\mathrm{l}}(f)$ are defined for local output at processor P_{i_0} in analogy to the case of global output.

Remark 2.9. While restriction $(*)^1$ (respectively $(*)^2$ in the 2-way case) may not seem to be a natural assumption to make in fully synchronous networks, it arises quite naturally in connection with the problem of performing synchronous algorithms on fully asynchronous networks. Assume that in an asynchronous network internal computations of processors and delivery of messages may be delayed for indefinite but finite periods of time. Still, we want to perform an algorithm written for a synchronous network in which processors can either receive or send one message in one step. The obvious idea is to have each processor keep an internal (virtual) step counter, and to keep the exchange of messages synchronized by the use of time stamps. Conceptually,

we assume that there is a "communication system" that manages a global message buffer. If P_i wants to send a message $m_{i,t}$ to one of the processors P_j, $j \in V_{i,t}$, it places the message, extended by a time stamp t, and the list $V_{i,t}$ into this buffer, and proceeds to step $t+1$ no matter what happens to the message. If P_j wants to receive a message in step t from one of the processors P_i, $i \in W_{j,t}$, it submits a corresponding request to the system, again with time stamp t. The system searches the buffer for a message that has time stamp t and lists P_j among its possible recipients. If one is found, it is delivered to P_j and removed from the buffer; P_j proceeds to step $t+1$. Otherwise, P_j is blocked until a message appears in the buffer that can be delivered to it. Synchronous algorithms that obey restriction $(*)^1$ are suited for being run in this way on an asynchronous network, in the following sense: no matter in which order messages are submitted or delivered, every receive request for step t will finally be satisfied; it never happens that a processor waits indefinitely for a message that will not arrive. (In the terminology of asynchronous distributed systems, this is a "liveness" property; see [30].)

2.3. Complexity measures for functions

We recall some definitions concerning functions $f : A_1 \times \cdots \times A_n \to A$. Originally, these notions were formulated for Boolean functions, but they can readily be generalized to other domains (cf. [12]). The concept of a *critical input* will be central for the main result. For discussion and applications of all these notions in the context of parallel computing on PRAMs see, e.g., [6,7,26,31], and in particular the survey [12]. To help readers to familiarize themselves with these concepts, we provide some simple examples.

Definition 2.10. Fix a function $f : A_1 \times \cdots \times A_n \to A$.

(a) An input $a = (a_1, \ldots, a_n)$ is *critical* for f if for every $i \in \{1, \ldots, n\}$ there is an input b that differs from a only in the ith component and satisfies $f(a) \neq f(b)$.
(b) The *critical complexity* $c(f)$ is the maximal k such that there is an input a with the property that for k different indices $i \in \{1, \ldots, n\}$ there is an input b that differs from a only in the ith component and satisfies $f(a) \neq f(b)$.
(c) For each input a, the *sensitive complexity* (or *certificate complexity*) $s(f, a)$ of f at a is the minimal k such that there is a set $I_a \subseteq \{1, \ldots, n\}$ of cardinality k with the property that all inputs $b = (b_1, \ldots, b_n)$ with $\forall i \in I_a$: $a_i = b_i$ satisfy $f(a) = f(b)$. The sensitive complexity $s(f)$ of f is $\max\{s(f, a) \mid a \text{ is an input}\}$.
(d) The function f *depends* on input bit i if there are inputs a and b that differ only in the ith component and satisfy $f(a) \neq f(b)$.

Example 2.11. (a) Input $(0, \ldots, 0)$ is critical for the n-ary OR function, but no other input is critical. For the n-ary PARITY function every input is critical.

(b) The critical complexity of the n-ary OR function is n; the critical complexity of the function h defined in Section 1.2 is $m = \sqrt{n}$.

(c) If $a=(0,\ldots,0)$ and $a'=(1,0,\ldots,0)$, then $s(\mathrm{OR},a)=n$ and $s(\mathrm{OR},a')=1$. The sensitive complexity of the n-ary OR function is n. The sensitive complexity of the function h from Section 1.2 is $m=\sqrt{n}$. (For each input a, it is sufficient to fix m input bits to guarantee that the output is $h(a)$.)

(d) All functions mentioned in (a)–(c) depend on all their input bits.

3. Computing functions versus gossiping

This section contains the formulation and the proof of the main results concerning the relationship between gossip and accumulation protocols on the one hand and computing functions in networks with global respectively local output on the other hand.

Throughout this section, let $G=(V,E)$ be a graph with $V=\{1,\ldots,n\}$ for some $n\in\mathbf{N}$, and let $i_0\in V$ be fixed. We assume alternately that G represents a gossiping network and a processor network. The following is well known and an almost immediate consequence of the definitions.

Observation 3.1. *If $f:A_1\times\cdots\times A_n\to A$ is an arbitrary n-ary function, then*

(a) $T_G^{1,\mathrm{g}}(f)\leqslant r(G)$,
(b) $T_G^{2,\mathrm{g}}(f)\leqslant r_2(G)$,
(c) $T_{G,i_0}^{1,1}(f)$, $T_{G,i_0}^{2,1}(f)\leqslant b(G,i_0)$, *for* $i_0\in V$.

If f is the OR *of n bits, it can be computed within these time bounds by using* 1*-bit messages only.*

Proof. We only consider (a); the other statements are proved similarly. Let $\mathscr{M}=(M_1,\ldots,M_T)$ be a 1-way gossip protocol, and let $a=(a_1,\ldots,a_n)$ be the input. The sequence $\mathscr{K}(\mathscr{M})=(K_0,K_1,\ldots,K_T)$ is defined as in Definition 2.3. In step t, $1\leqslant t\leqslant T$, the processors send each other messages according to the communication pattern given by M_t. If $(i,j)\in M_t$, processor P_i sends a message $m_{i,t}$ to P_j, where $m_{i,t}$ is a representation of the input fragment $(a_l)_{l\in K_i(t-1)}$. From the definition of $\mathscr{K}(\mathscr{M})$ it follows by induction that the required information is available to P_i in this step; since $\mathscr{M}$ is a gossip protocol, after step T each processor knows the complete input a and can compute $f(a)$. Note that the longest message can be at most as long as a. If f is the OR function, it is sufficient to send as message $m_{i,t}$ the single bit $\bigvee_{l\in K_i(t-1)}a_l$. (Note that the only properties of the OR function used here are that it is commutative, associative, and idempotent. In complete networks, idempotency is not required, see [2,5].) □

The main result of this paper is essentially that, under the restriction "predictable reception", these algorithms for computing f are optimal if global respectively local output is required and f has a critical input. If global output is required and f is nonconstant, lower bounds for broadcast protocols apply.

Theorem 3.2. *Let $G=(V,E)$ be a graph with node set $V=\{1,\ldots,n\}$, which represents a network of processors, and let f be an n-ary function. If f has a critical input, then*

(a) $T_G^{1,g}(f)=r(G)$,
(b) $T_G^{2,g}(f)=r_2(G)$, *and*
(c) $T_{G,i_0}^{1,1}(f)=T_{G,i_0}^{2,1}(f)=b(G,i_0)$, for $i_0\in V$.

Further,

(d) *if f depends on the i_0th input component for some $i_0\in V$, then $T_G^{1,g}(f)$, $T_G^{2,g}(f)\geqslant b(G,i_0)$.*

Corollary 3.3. *The complexity of computing the* OR *of n bits on G (under restriction $(*)^1$ resp. $(*)^2$) is given by*

(a) $r(G)$ $[r_2(G)]$ *for* 1*-way* [2*-way*] *communication and global output, and*
(b) $b(G,i_0)$ *for* 1*-way or* 2*-way communication and local output at processor P_{i_0}.*

(*In the upper bounds*, 1*-bit messages are sufficient*.)

Proof of Theorem 3.2. We deal with part (a) in detail; the proofs of the other parts, being similar, will only be sketched. In view of Observation 3.1, only the inequality $T_G^{1,g}(f)\geqslant r(G)$ has to be proved. For this, assume that a 1-way algorithm for computing f on G in T steps is given, and that $a^*=(a_1^*,\ldots,a_n^*)$ is a critical input for f. Obviously, it is sufficient to construct a 1-way gossip protocol for G that has T rounds.

The construction splits into two parts. First, we eliminate ambiguities from computations according to the algorithm, i.e., for each input a we fix a computation C_a, which essentially corresponds to a communication pattern. In the second part, we show that C_{a^*} induces a gossip protocol for G.

Part 1: *Fix computations*. Consider the network algorithm that computes f in T steps. First, we arbitrarily fix a computation C_{a^*} for input a^*. Consider steps $t=1,\ldots,T$ one after the other. Let $E_t(a^*)$ be the set of all possible pairs of senders and recipients determined on the basis of step $t-1$ (cf. Section 2.2). Then an arbitrary matching $M_t^*\subseteq E_t(a^*)$ is chosen that covers all recipients, and messages are delivered according to M_t^*. Now, consider some input $a\neq a^*$. We proceed by induction on t. Assume C_a has been fixed up to step $t-1$, and consider the graph $(V,E_t(a))$ induced by the communication requests of the processors for step t. Let $M_t'(a)=M_t^*\cap E_t(a)$ be the set of those edges in $E_t(a)$ that are used in step t of C_{a^*}. By restriction $(*)^1$, we may extend $M_t'(a)$ to some directed matching $M_t(a)\subseteq E_t(a)$ that covers all recipients. Deliver messages according to $M_t(a)$.

Part 2: *Identify a gossip protocol*. Consider the communication protocol $\mathscr{M}^*=(M_1^*,\ldots,M_T^*)$. The proof of part (a) of the theorem is finished once the following assertion is established.

Lemma 3.4 (Main lemma). *$\mathscr{M}^*$ is a 1-way gossip protocol for G.*

Proof. Let the sequence $\mathscr{K}(\mathscr{M}^*) = (K_0, K_1, \ldots, K_T)$ be as in Definition 2.3. We must show that $K_T(i) = V$ for all $i \in V$. For this, it is sufficient to establish the following assertion $(\mathscr{A}_t)$ for $t = T$:

$(\mathscr{A}_t)$ for all inputs $a = (a_1, \ldots, a_n)$ and all $i \in V$:
$(\forall j \in K_t(i) : a_j = a_j^*) \Rightarrow P_i$ is in the same state in C_{a^*} and C_a after step t.

Indeed, if $K_T(i) \neq V$, e.g., $j \notin K_T(i)$, then by $(\mathscr{A}_T)$ processor P_i is in the same state after step T on input a^* and each input that differs from a^* only in the jth component, which contradicts the assumption that a^* is critical for f and that the network computes the function f with global output.

We prove $(\mathscr{A}_t)$ by induction on t. For $t = 0$, the claim follows from the definitions: $K_0(i) = \{i\}$, for $i \in V$, and the initial state of P_i only depends on the ith input component. Now assume $t > 0$, and that $(\mathscr{A}_{t-1})$ is true. Let $i \in V$ and a be an input so that $a_j = a_j^*$ for all $j \in K_t(i)$. There are two cases.

Case 1: There is no l such that $(l, i) \in M_t^*$. Then, by definition, $K_t(i) = K_{t-1}(i)$, in particular $a_j = a_j^*$ for all $j \in K_{t-1}(i)$. By the induction hypothesis, P_i is in the same state after step $t-1$ in both C_{a^*} and C_a. Since no l satisfies $(l, i) \in M_t^*$, no processor P_l delivers a message to P_i in step t of C_{a^*}, i.e., by restriction $(*)^1$ ("predictable reception"), in step t of C_{a^*} processor P_i is a sender. Since this decision is based on the state at the end of step $t-1$, P_i decides in the same way in computation C_a, thus does not receive a message in C_a in step t either, and enters the same state in C_a as in C_{a^*}.

Case 2: There is some (unique) l such that $(l, i) \in M_t^*$. In this case, by definition, $K_t(i) = K_{t-1}(l) \cup K_{t-1}(i)$. We apply the induction hypothesis $(\mathscr{A}_{t-1})$ to a with respect to both $K_{t-1}(l)$ and $K_{t-1}(i)$ to conclude that P_l is in the same state after step $t-1$ of C_a and of C_{a^*}, and that the same is true for P_i. Since $(l, i) \in M_t^*$, the message $m_{l,t}$ sent by processor P_l in this step is delivered to P_i in step t of computation C_{a^*}; in particular, P_l is a sender with $i \in V_{l,t}$ and P_i is a recipient with $l \in W_{i,t}$. Since both P_i and P_l are in the same state after step $t-1$ in C_{a^*} and C_a, this will also be true in computation C_a; thus, edge (l, i) is in $E_t(a)$ and in M_t^*, and P_l sends message $m_{l,t}$ in step t of C_a as well. By the construction of C_a described in Part 1, edge (l, i) will be chosen to be in $M_t(a)$, and message $m_{l,t}$ will be delivered to P_i in C_a. This implies that P_i receives identical messages in step t of C_{a^*} and of C_a; hence P_i will enter the same state at the end of step t in these two computations. This finishes the induction step, and the proof of the main lemma. □

We will not prove parts (b) and (c) of the theorem in detail, since the arguments are essentially the same as in part (a). We only point out the changes to be made. In part (b), undirected matchings are used in place of directed matchings. We fix a computation C_{a^*} arbitrarily, and show exactly as in the 1-way case that the sequence $\mathscr{M}^* = (M_1(a^*), \ldots, M_T(a^*))$ of matchings that describe the pairs of processors that communicate in each step is a 2-way gossip protocol. Restriction $(*)^2$ is formulated

in such a way that the argument from (a) carries over without any difficulties. In part (c), the construction of the communication protocol is exactly the same as in part (a), respectively (b). The only difference in the proof is that since it is only required that P_{i_0} knows the result at the end, we can only conclude that $K_T(i_0) = V$, which means that we have constructed an accumulation protocol for G. Then Fact 2.6 and the remark preceding it yield (c).

Finally, we sketch the proof of part (d), for 1-way mode. Choose a^* and a such that $f(a^*) \neq f(a)$ and a^* and a differ only in the i_0th component. Computation C_{a^*} and (afterwards) computation C_a, each with T steps, are fixed exactly as in the proof of part (a); the sequence $\mathscr{M} = (M_1(a^*), \ldots, M_T(a^*))$ is defined as above. Consider the resulting sequence $\mathscr{K}(\mathscr{M}) = (K_0, \ldots, K_T)$ (Definition 2.3). The assertion

$(\mathscr{A}'_t) \quad \forall i \in V [i_0 \notin K_t(i) \;\Rightarrow\; P_i$ is in the same state after step t of C_{a^*} and $C_a]$

is proved by induction on t. Since all processors P_i must be able to distinguish a^* and a after step T, we must have $i_0 \in K_T(i)$ for all $i \in V$, thus, $\mathscr{M}^*$ is a broadcast protocol, which establishes the inequality $T \geqslant b(G, i_0)$. □

4. Extensions and limitations

In this section, we consider possible extensions of the main result. First, we discuss functions that do not necessarily have a critical input; next, multiple-output functions are considered; finally, the situation of processor networks that do not necessarily satisfy restriction (∗) is discussed. The complexity of broadcasting a bit in networks with an even more general communication mode is treated in detail in Section 5.

4.1. Functions without critical input

What can be said about functions that do not have a critical input? We can use the approach taken in the proof of Theorem 3.2 to obtain the following. Assume a network $G=(V,E)$ computes a function f with global output, and $a^*=(a_1^*, \ldots, a_n^*)$ is an arbitrary input. Define $\mathscr{M}^* = (M_1^*, \ldots, M_T^*)$ as before, in accordance with an arbitrarily chosen computation on this input. Then it is easily seen that assertion $(\mathscr{A}_T)$ still holds: for all inputs a and all $j \in V$ we have that if $a_i = a_i^*$ for all $i \in K_T(j)$, then $f(a) = f(a^*)$. This means that for each processor P_j the set $K_T(j)$ must be a "certificate set" I_{a^*} for the function value $f(a^*)$, as in Definition 2.10(c). If we choose a^* to be an input with $s(f) = s(f, a^*)$, we obtain that $|K_T(j)| \geqslant s(f)$ for all $j \in V$.

Definition 4.1. For $1 \leqslant s \leqslant n$, let $r(G,s)$ denote the minimum number of rounds of a communication protocol for G that satisfies $|K_T(j)| \geqslant s$ for all $j \in V$.

While the message complexity of the problem to obtain at least s different pieces of information was defined and investigated already in [27], its round complexity $r(G,s)$ does no seem to have received much attention, with the following exception: in the case

of the complete network, in [4] it was established that the technique of [11,22,23,28] can quite easily be adapted to prove a lower bound of about $\log_\rho(s)$ for this case with $\rho = \frac{1}{2}(1+\sqrt{5})$, which is about $1.44\ldots\log(s)$.

Alternatively, one can consider the critical complexity of f (Definition 2.10(b)). Let a^* be an input and $I \subseteq V$ be a set of size $c(f)$ such that for every $i \in I$ there is an input a that differs from a^* only in the ith component and satisfies $f(a) \neq f(a^*)$. Then from $(\mathscr{A}_T)$ it is clear that the communication protocol defined in the proof of Theorem 3.2 satisfies $I \subseteq K_T(j)$ for all $j \in V$. This relates to the problem of broadcasting from $k = c(f)$ fixed sources. Results for this problem for specific networks and specific placement of these sources have been obtained by Höltring in her diploma thesis [17].

4.2. Functions with multiple outputs

In applications, often functions $f : A_1 \times \cdots \times A_n \to B_1 \times \cdots \times B_n$, $a \mapsto (f_1(a), \ldots, f_n(a))$, must be computed in such a way that the jth component $f_j(a)$ appears at processor P_j at the end. In [8], matrix inversion, discrete Fourier transform, and sorting are listed as examples. Our methods apply to this situation *directly* if there is one input a^* that is critical for all functions $f_1, \ldots, f_n$. This is the case, e.g., for the problem of sorting n numbers $a_1, \ldots, a_n$ from the set $\{1, \ldots, m\}$ with $m \geqslant n+2$, so that the number with rank j appears at processor P_j at the end, since the input $(2, \ldots, n+1)$ is critical for all output positions. In many important cases, though, no single input can be found that is critical for all components of the output, e.g., for the problem of sorting n bits. For the special case of the complete network, we still can obtain lower bounds in terms of the sensitive complexity of $f_1, \ldots, f_n$ on specific inputs, in the following way.

Proposition 4.2. *If the multiple-output function $f : a \mapsto (f_1(a), \ldots, f_n(a))$ is computed on the complete network in 1-way mode in T steps, under restriction $(*)$[1], and a^* is any input, then*

$$T \geqslant \frac{1}{2} \cdot \log_\rho \left(\left(\sum_{1 \leqslant j \leqslant n} s(f_j, a^*)^2 \right) \Big/ n \right)$$

$$\geqslant 0.72\ldots \cdot \log \left(\left(\sum_{1 \leqslant j \leqslant n} s(f_j, a^*)^2 \right) \Big/ n \right),$$

where $\rho = (1+\sqrt{5})/2$.

Proof. (Sketch.) Fix any input a^*. Just as in Section 4.1 we can see that the communication protocol $\mathscr{M}^*$ induced by a^* and the corresponding sequence $\mathscr{K}(\mathscr{M}^*) = (K_1, \ldots, K_T)$ must satisfy $|K_T(j)| \geqslant s(f_j, a^*)$, for $1 \leqslant j \leqslant n$. The proof method from

[8,12,23,28] can be applied to show that T must satisfy

$$\rho^T \cdot \sqrt{n} \geqslant \left(\sum_{1 \leqslant j \leqslant n} |K_T(j)|^2 \right)^{1/2}.$$

Combining these inequalities and taking logarithms yields the result. □

Corollary 4.3. *Sorting n bits in* 1*-way mode on a complete network with restriction* $(*)^1$ *takes at least* $1.44\ldots \log n - \mathrm{O}(1)$ *steps.*

Proof. Assume the problem of sorting n bits can be solved in T steps. The function f_j is the output bit that appears at processor P_j. As input, we choose $a^* = (0,\ldots,0)$. Since $f_j(a) = 0$ if and only if a contains j zeroes, it is clear that $s(f_j, a^*) = j$. By Proposition 4.2, $T \geqslant \frac{1}{2} \log_\rho((\sum_{1 \leqslant j \leqslant n} j^2)/n) = \log_\rho(n) - \mathrm{O}(1)$. □

We leave it to the reader to find inputs for the other problems mentioned (discrete Fourier transform or matrix inversion) that make it possible to show that these problems have high complexity in complete processor networks.

In order to apply results from the gossiping framework to the computation of multiple-output functions in other, sparse, networks, we would have to have more information on partial gossiping in these networks, in particular, the complexity of collecting any k input pieces or a predetermined set of k input pieces in every node.

4.3. Networks without "predictable restriction"

In this section, we show that Theorem 3.2 becomes false if the network algorithm does not have to satisfy restriction $(*)$ ("predictable restriction") but still is fully synchronous. In this case, we may collect information with a large fan-in, and distribute information faster using the phenomenon of "transmitting information by *not* sending a message", cf. [3,7,9]. Then, we show that even such networks cannot compute non-constant functions with global output more than four times faster than gossip protocols.

Let us start with two examples, the tree network and the complete network.

Proposition 4.4. *Let $d \geqslant 1$, and let G be a full binary tree of depth d with root P_1.* (*Thus, G has $2^{d+1} - 1$ nodes.*) *With synchronous algorithms in* 1*-way mode* (*which do not satisfy restriction* $(*)^1$), *in G*:

(a) *the* OR *of n bits with local output at P_1 can be computed in d steps,*
(b) *a bit b located initially at P_1 can be broadcasted to all nodes in d steps, and*
(c) *the* OR *of n bits with global output can be computed in $2d$ steps.*

Proof. (a) It is sufficient to note that (for $d = 1$) the OR of three bits a_1, a_2, and a_3 can be computed in 1 step, because then we can proceed iteratively, starting from the leaves. In one step, P_2 [P_3] sends a message to P_1 if and only if $a_2 = 1$ [$a_3 = 1$]. If P_1 gets any messages, the result is 1, otherwise it is equal to a_1. (b) Again, we only

have to show how this can be done for $d = 1$; in larger trees, we iterate, starting from the root. P_1 informs both of its sons in one step as follows: if $b = 0$ then it sends a message to P_2; if $b = 1$ then it sends a message to P_3. In each case, the processor that did not get a message can derive b from just this fact. (The contents of the message are irrelevant; cf. [3,10] for this trick.) (c) Apply (a) and (b) one after the other. □

The resulting running times should be contrasted with the fact that if G is the binary tree of depth d with root P_1 then $b(G, P_1) = 2d$ and $r(G) = 4d$ (cf. [19]). In the complete network, the situation is slightly different.

Proposition 4.5. *Let G be the complete network of size n. With synchronous algorithms in* 1*-way mode* (*which do not satisfy restriction* $(*)^1$), *in G*:

(a) *the* OR *of n bits with local output can be computed in one step*,
(b) *a bit c located initially at some node can be broadcasted to all nodes in* $\lceil \log_3 n \rceil \approx 0.63 \log n$ *steps*, *and*
(c) *the* OR *of n bits with global output can be computed in* $1 + \lceil \log_3 n \rceil$ *steps*.

Proof.

(a) For $2 \leqslant i \leqslant n$, processor P_i sends a message to P_1 if and only if $a_i = 1$. If P_1 gets a message, the result is 1, otherwise it is a_1.
(b) Using the same trick as in the proof of the previous proposition, in one step a processor that knows c can inform two other processors of this value; thus, the number of processors that know c can be tripled in one step.
(c) Combine (a) and (b). □

The running times from this proposition should be compared with the values $a(G, i_0) = b(G, i_0) = \lceil \log n \rceil$ and $r(G) = 1.44 \ldots \log n \pm \mathrm{O}(1)$ for G the complete network. Next, we note that for computing nonconstant functions with global output in any network G, at least $\frac{1}{4} r(G)$ steps are needed.

Proposition 4.6. *If a nonconstant function f can be computed in T steps with global output in the network G* (*without restriction* $(*)^1$), *then* $b(G, i_0) \leqslant 2T$ *for some* i_0, *and* $r(G) \leqslant 4T$.

Proof. The main part of the proof is postponed to Section 5, where we characterize the complexity of broadcasting a bit in synchronous networks that are even stronger than those considered here in terms of a variant of the broadcast complexity of the underlying graph. Clearly, if a network G can compute a function f that depends on input position i_0 in T steps with global output, then it can broadcast one bit from source P_{i_0} in T steps. In Section 5, we show that broadcasting one bit from a node i_0 in G takes at least $b_2(G, i_0)$ steps, which is the number of rounds in an optimal "2-broadcast" protocol, in which one node may send a message to two of its neighbors

in one step. It is easy to see that $b(G,i_0) \leqslant 2b_2(G,i_0)$, hence $b(G,i_0) \leqslant 2T$. By the trivial fact that $r(G) \leqslant 2b(G,i_0)$ (cf. [19]), we obtain $r(G) \leqslant 4T$. □

We note that the bound given in the previous proposition is tight: Let G be the binary tree of depth d. We can broadcast a bit from the root in d steps (Proposition 4.4(b)), hence in G some nonconstant function can be computed with global output in d steps. On the other hand, $r(G) = 4d$ for this network, and $b(G,i_0) \geqslant 2d$ for every node i_0 in G.

5. Broadcasting a bit in a synchronous network

In this section, we study the complexity of broadcasting one bit in a synchronous network of processors that communicate by message passing, with hardly any restriction on the communication mode excepting that a processor must not send more than one message in one step. We show, in analogy to Theorem 3.2, that a well-known complexity measure from the theory of broadcast protocols for atomic pieces of information is appropriate for describing the complexity of this problem.

Note that the problem of broadcasting a bit captures the essence of all situations in which differences in the state of a single processor finally influence all others.

We treat communication in 1-way mode and in 2-way mode separately (Sections 5.3 and 5.4, respectively). The results of this section can be applied to other parallel models like EXCLUSIVE READ PRAMs and 1-ARBITRARY distributed memory machines, as will be indicated at the end of Section 5.4.

5.1. The general network model in 1-way mode

As before, we consider a network of n processors, connected by bidirectional links according to a graph $G = (V,E)$. First, we focus on 1-way communication. One node P_{i_0} is distinguished as the source of the broadcasting process, i.e., this processor can be in either one of two different states initially, representing inputs 0 and 1; the other processors are in an initial state that is independent of the input. In step t, a processor may send a message *and* receive several messages, as specified by the following rules. Depending on its state after step $t-1$, processor P_i fixes a message $m_{i,t}$ that it wishes to send, and a set $V_{i,t} \subseteq \{j \mid \{i,j\} \in E\}$ of possible recipients. (As in Section 2.2, the choice $V_{i,t} = \emptyset$ indicates that P_i does not send a message at all; with $|V_{i,t}| = 1$ a unique recipient can be specified.) Further, P_i specifies a set $W_{i,t} \subseteq \{j \mid \{i,j\} \in E\}$ of processors from which it wishes to receive a message. Let $E_t = \{(i,j) \mid j \in V_{i,t} \text{ and } i \in W_{j,t}\}$. Some set $E'_t \subseteq E_t$ is selected arbitrarily such that for each i that occurs as a first component in E_t there is exactly one j such that $(i,j) \in E'_t$. Then, for each $(i,j) \in E'_t$, message $m_{i,t}$ is delivered to P_j. Thus, a processor P_j that has specified $W_{j,t} \neq \emptyset$ may receive messages from none, some, or all P_i with $j \in W_{j,t}$. On the basis of *all* messages received P_j changes its state. After step T, all processors must know whether the input bit was 0 or 1. (The reader is invited to check against his or her intuition that alternative rules for

dealing with surplus messages like buffering for later delivery, combining, discarding, etc. are at most as strong as this scheme.)

Definition 5.1. The *broadcast complexity* $\hat{T}_{G,i_0}$ of a processor network G in 1-way mode with source P_{i_0} is the smallest number of steps an algorithm for broadcasting a bit from P_{i_0} can have.

5.2. 2-broadcast protocols

Turning now to communication networks, we generalize broadcast protocols (see Section 2) to 2-*broadcast protocols*, in which one node may pass the information it has to two of its neighbors in one step. (This modification is discussed as "DMA-bound model" $H2$, an abbreviation for "half-duplex with outdegree 2", in [13,22].) Our definition of 2-broadcast protocols differs only formally from the standard definition.

Definition 5.2. Let $G=(V,E)$ be a graph, and $v_0 \in V$.

(a) A 2-broadcast protocol in 1-way mode with source node i_0 is a sequence $\mathscr{M} = (M_1,\ldots,M_T)$ of sets M_t of directed edges with the following properties:
  (i) if $(i,j) \in M_t$ then $\{i,j\} \in E$, for $1 \leqslant t \leqslant T$,
  (ii) i occurs at most twice as a first component in M_t and at most once as a second component in M_t, for each $i \in V$ and $1 \leqslant t \leqslant T$, and
  (iii) if $\mathscr{K}(\mathscr{M}) = (K_1,\ldots,K_T)$ is as in Definition 2.3 then $i_0 \in K_T(i)$ for all $i \in V$.

(b) The 2-broadcast complexity of G with source node i_0 is the minimum T such that there is a 2-broadcast protocol as in (a) with T rounds.

A 2-broadcast protocol can be used to broadcast a piece of information, initially located at node i_0, in the following obvious way: if edge (i,j) is in M_t and $i_0 \in K_t(i)$, then in round t the piece of information is sent from i to j. The definition makes sure that the node that has to send the piece of information actually has received it before and that every processor sends the piece of information to at most two of its neighbors in one step. As mentioned before, it is obvious that $b(G,i_0) \leqslant 2b_2(G,i_0)$ for arbitrary networks G and nodes i_0 in G.

5.3. Processor networks versus 2-broadcast networks

By using the trick described in the proof of Proposition 4.4(b) for sending one bit to two recipients in one step, we can transform any 2-broadcast protocol for a graph G into a broadcast algorithm in 1-way mode for the processor network with topology given by G.

Observation 5.3. $\hat{T}_{G,i_0} \leqslant b_2(G,i_0)$.

Proof. Assume that a 2-broadcast protocol for G with T rounds is given, and define $\mathscr{K}(\mathscr{M}) = (M_1,\ldots,M_T)$ as before. We obtain an algorithm for the network as follows. If

edges (i,j_1) and (i,j_2) are in E_t and $i_0 \in K_t(i)$, then in step t processor P_i informs P_{j_1} and P_{j_2} of the value of b with the trick described in the proof of Proposition 4.4(b). If there is only one edge leaving i in E_t, the trick can be applied nonetheless. We note that in this way all processors can use the same message in all steps. □

The purpose of this section is to show that this algorithm is optimal. This may be intuitively plausible, but to the best of the knowledge of the author a formal proof, especially for a network model as general as considered here, has not been available so far.

Theorem 5.4. $\hat{T}_{G,i_0} = b_2(G,i_0)$.

Proof. In view of Observation 5.3 we must only prove that $\hat{T}_{G,i_0} \geqslant b_2(G,i_0)$. Let a broadcast algorithm for the processor network G in 1-way mode with source i_0 be given that runs in T steps. We construct a 2-broadcast protocol with at most T steps.

We proceed similarly as in the proof of Theorem 3.2. However, here computations C_0 and C_1 on inputs 0 and 1 are fixed simultaneously by induction on t. The idea is to choose the actual transmissions for C_0 and C_1 in such a way that they have as many common elements as possible.

Thus, assume that C_0 and C_1 have been fixed up to step $t-1$. For $b=0,1$, the communication requests of the processors (as represented by the sets $V_{i,t}(b)$ and $W_{i,t}(b)$, for $i \in V$) induce a set $E_t(b)$ of directed edges across which messages may flow. We let $E_t'' = E_t(0) \cap E_t(1)$ and choose a subset $E_t' \subseteq E_t''$ so that if i appears as a first component in E_t'' then $(i,j) \in E_t'$ for exactly one j. Then, for $b=0,1$ separately, a set $E_t'(b)$ with $E_t' \subseteq E_t'(b) \subseteq E_t(b)$ is chosen with the property that if i appears as a first component in $E_t(b)$ then $(i,j) \in E_t'(b)$ for exactly one j, and in step t of C_b message $m_{i,t}$ is delivered to P_j for all pairs $(i,j) \in E_t'(b)$.

The intuition behind the next definition is that we try to "peel off" inessential parts of the two computations, i.e., to identify "meaningless" messages and eliminate them. For this, we define a set $E_{0,1}$ of labeled, directed edges that run along some of the edges of G, and identify a 2-broadcast protocol as part of this set.

- Edge (i,j), with label t, for $1 \leqslant t \leqslant T$, is in $E_{0,1}$ if and only if there is some $b \in \{0,1\}$ such that in computation C_b the message $m_{i,t}$ sent by processor P_i is delivered to P_j, but in the other computation $C_{\bar{b}}$ no message or a different one is sent from P_i to P_j in step t.

Note that parallel edges are possible, involving different time steps. Note also that identical messages that are sent across the same edge in both C_0 and C_1 are ignored in this definition, corresponding to the intuition that they are "meaningless". The following simple observations are crucial.

Claim 1. Let $i \neq i_0$ and $j \neq i_0$. If (i,j) with label t is in $E_{0,1}$, and no edge (i',j) with a label $t' < t$ is in $E_{0,1}$, then there must be some $l \in V$ such that edge (l,i) with label t'' is in $E_{0,1}$ for some $t'' < t$.

(Intuitively spoken, in order to send a meaningful message in step t, a processor P_i with $i \neq i_0$ must either

- have received a meaningful message in an earlier step in the same computation, or
- have noticed that a meaningful message that would have been due to arrive in the other computation has not turned up, or
- be treated differently by the intended recipient of the message in the two computations.)

Claim 2. If $i \neq i_0$ then there is at least one labeled edge that enters node i.

(Intuitively, if P_i, $i \neq i_0$, never gets a meaningful message in either computation, it cannot know the input bit at the end.)

Proof of Claim 1. Assume for a contradiction that (i,j) with label t is in $E_{0,1}$, but that in $E_{0,1}$ there is neither an edge (i',j) nor an edge (l,i) with a label $t'' < t$. By symmetry, we may assume without loss of generality that in computation C_0 processor P_i sends a message $m_{i,t}$, which is delivered to P_j, but that this message does not flow from P_i to P_j in C_1. This means that $(i,j) \in E_t(0)$, hence we have $j \in V_{i,t}(0)$ and $i \in W_{j,t}(0)$. The last two assumptions imply that processor P_i, which does not know b initially since $i \neq i_0$, has received exactly the same messages in each of the steps $1,\ldots,t-1$ in both computations, and that the same is true for P_j. This entails that P_i and P_j both are in the same state after step $t-1$ in C_0 and C_1, hence $j \in V_{i,t}(1)$ and $i \in W_{j,t}(1)$, which implies $(i,j) \in E_t(1)$, and P_i sends message $m_{i,t}$ in C_1 as well. By the definition of E''_t, edge (i,j) appears in this set. In computation C_0 edge (i,j) is used for delivering $m_{i,t}$, which implies $(i,j) \in E'_t$. Hence, $(i,j) \in E'_t(1)$ as well, which means that message $m_{i,t}$ is delivered to P_j in C_1 as well, a contradiction.

Proof of Claim 2. If no labeled edges enter i, then all messages received by P_i in steps t, $1 \leqslant t \leqslant T$, are identical in C_0 and C_1. Thus, since $i \neq i_0$, in both computations P_i will be in the same state at the end of step T. This contradicts the requirement that at the end all processors must know b.

Next, we eliminate some of the labeled edges, as follows. All labeled edges that enter i_0 are removed; all labeled edges that enter $i \neq i_0$ excepting that one with the smallest label t are also removed. (This operation corresponds to the intuition that once a processor has received a meaningful message, or noted that it did not arrive at its due time, it knows b, and later messages are irrelevant.) The resulting edge set is called $E^*_{0,1}$. The digraph $G^* = (V, E^*_{0,1})$ has the following two properties:

(i) All nodes excepting i_0 have indegree 1 in G^*.
(ii) If (i,j) with label t is in $E^*_{0,1}$, then either $i = i_0$ or the (unique) edge that enters i in G^* has a label $t' < t$.

(Property (i) is immediate from Claim 2 and the definition of $E^*_{0,1}$. Property (ii) is a consequence of Claim 1 and the definition of $E^*_{0,1}$. Indeed, if (i,j) with label t is

in $E^*_{0,1}$, then there cannot be an edge (i',j) in $E_{0,1}$ with an "earlier" label $t' < t$, and $j \neq i_0$. We may assume that $i \neq i_0$, and apply Claim 1 to conclude that there is an edge in $E_{0,1}$ that enters i and has a label $t'' < t$. Again by the definition of $E^*_{0,1}$, the label of the unique edge $(l,i) \in E^*_{0,1}$ that enters i must have a step number which is at most t'', thus, smaller than t.)

Properties (i) and (ii) taken together say that G^* is a directed spanning tree for V with root i_0, and that the labels along directed paths in G^* are strictly increasing with respect to their t-parts. Moreover, due to the definition of $E_{0,1}$, for each node and each t there can be at most two edges leaving i that are labeled with t. From this it easily follows that we obtain a 2-broadcast protocol for G with at most T rounds by defining $E_t = \{(i,j) \mid (i,j) \in E^*_{0,1} \text{ and } (i,j) \text{ has label } t\}$, for $1 \leqslant t \leqslant T$. □

5.4. Broadcasting in 2-way mode and in other parallel models

The complexity of broadcasting a bit on EREW PRAMs has been determined in [3]. Let us recall briefly how such a parallel model works. It consists of processors $Q_1, \ldots, Q_p$ and cells $C_1, \ldots, C_r$. In one step, a processor reads from a cell, performs some internal computation, and writes to a cell. It is forbidden that in any step more than one processor reads from the same cell or more than one processor writes to the same cell. Initially, one cell contains a bit, the others have a neutral content. EREW algorithms can be regarded as algorithms for processor networks with $n = p + r$ processors. Each PRAM processor and each PRAM cell are represented as a network processor. A write phase is represented as a step in which all PRAM processors are senders (with 0 or 1 possible recipients) and all cells are recipients (specifying all PRAM processors as possible senders). A read phase is represented as a step in which all PRAM processors are recipients (specifying 0 or 1 cell as possible sender). If we apply the technique from the previous section, we obtain a 2-broadcast tree in which levels alternately consist of PRAM cells and PRAM processors. It is not hard to show that the numbers q_t of the cumulative size of the levels 1 through t of processors and the numbers c_t of the cumulative size of the level 1 through t of cells satisfy the following recurrence inequalities:

$$c_0 = 1,$$

$$p_0 = 0,$$

$$p_t \leqslant p_{t-1} + c_{t-1},$$

$$c_t \leqslant c_{t-1} + 2p_t$$

(cf. [3]). This implies that for all p processors to be informed of the bit, $\log_{2+\sqrt{3}} p \pm O(1)$ steps must be made. Incidentally, we note that the EXCLUSIVE WRITE rule is not needed in this argument; rather concurrent writing with any conflict resolution rule may be permitted.

We want to generalize this lower bound to a PRAM in which concurrent read accesses or write accesses are not forbidden but rather are resolved by the ARBITRARY rule.

ARBITRARY READ: If in a step several processors try to read from the same cell, an arbitrary one of them is given the contents of the cell, the other ones receive a negative acknowledgement ("reading failed").

ARBITRARY WRITE: If in a step several processors write to the same cell, an arbitrary one of them succeeds *and* is given an acknowledgement ("writing successful"), the other ones are given a negative acknowledgement ("writing failed").

The reader should note that for reading this is an unusual rule in the context of PRAMs. It has only been proposed in the context of distributed memory machines (DMMs) as a relaxation of the COLLISION access rule, see [10,25]. The DMM with COLLISION access rule, also known as OCPC, has been unter intensive investigation in recent years, and it has been shown that such machines are very strong in a randomized setting; in particular they are able to perform routing and simulate PRAMs in sublogarithmic time [14,15,25]. Here, we show that the deterministic version of the model has a significant weakness: the elementary task of broadcasting one bit to p processors takes $\Omega(\log p)$ steps.

If we try to translate the rules for such ARBITRARY-READ ARBITRARY-WRITE PRAMs into communication rules for a network as above for the EREW PRAM, we note that this time we must take 2-way communication into account, because of the acknowledgements received by the processors that write. Thus, we must describe communication rules for processor networks without restriction $(*)^2$ but allowing full-duplex use of links in one step. The rules for step t in such a network are as follows. Depending on its state after step $t-1$, each processor P_i fixes a message $m_{i,t}$ and a set $V_{i,t} \subseteq \{j \mid \{i,j\} \in E\}$ of possible partners. Let $V_t = \{i \in V \mid V_{i,t} \neq \emptyset\}$ and $E_t = \{\{i,j\} \mid i \in V_{j,t} \text{ and } j \in V_{i,t}\}$, and consider the graph $G_t = (V_t, E_t)$. A maximal (i.e., not extendible) matching $M_t \subseteq E_t$ is chosen arbitrarily, and for each edge $\{i,j\} \in M_t$ message $m_{i,t}$ is delivered to P_j and $m_{j,t}$ is delivered to P_i. Processors that have received a message change their state based on this message; processors that have not received anything change their state based on this knowledge. At the beginning, i.e., before step 1, processor P_{i_0} knows the bit b; after step T, all processors must know bit b no matter in which way the sets M_t were chosen.

Definition 5.5. The 2-way broadcast complexity $\hat{T}^2_{G,i_0}$ of a processor network G with source P_{i_0} is the smallest number of steps a 2-way algorithm for broadcasting a bit from P_{i_0} can have.

Remark 5.6. In analogy to Remark 2.9 we note that this kind of algorithm can be performed on an asynchronous network without deadlocks and without processors waiting indefinitely for messages that will not arrive. As in the case of algorithms that obey restriction $(*)^2$, in its local step t processor P_i places its message $m_{i,t}$ and the list $V_{i,t}$ of its possible partners into a global buffer, together with a time stamp t. Whenever the buffer contains a matching pair of processors (i.e., $i \in V_{j,t}$ and $j \in V_{i,t}$), after some finite delay the system delivers the messages $m_{i,t}$ and $m_{j,t}$ and removes the communication requests. Ties are broken arbitrarily. Whenever it happens that P_i has submitted a list

$V_{i,t}$, but all P_j with $j \in V_{i,t}$ either have specified $V_{j,t} = \emptyset$ or have already found other partners for communicating in step t, the system informs P_i that its communication attempt in step t has failed and P_i proceeds to step $t+1$. It is easily seen that the set $\{\{i,j\} \mid P_i$ exchanges messages with P_j in step $t\}$ is a maximal matching M_t in E_t, as required.

The 1-way algorithm of Observation 5.3 can also be run in 2-way mode, since no concurrent writing is used. Thus, we have the following.

Observation 5.7. $\hat{T}^2_{G,i_0} \leqslant b_2(G,i_0)$. □

However, one can also prove the (again intuitively obvious) fact that for the broadcast problem 2-way communication does not help.

Theorem 5.8. $\hat{T}^2_{G,i_0} = b_2(G,i_0)$.

Proof (Sketch). We must only prove that $b_2(G,i_0) \leqslant \hat{T}^2_{G,i_0}$. Let a 2-way algorithm be given that performs broadcast from P_{i_0}. We proceed similarly as in the proof of Theorem 5.4, and only indicate the changes. First, we consider undirected edges. Computations C_0 and C_1 are fixed simultaneously by induction on t. If $E_t(b) = \{\{i,j\} \mid i \in V_{j,t}(b)$ and $j \in V_{i,t}(b)\}$, for $b \in \{0,1\}$, is the collection of possible pairs of processors to communicate in step t, we first fix a maximal matching E'_t in $E''_t = E_t(0) \cap E_t(1)$ and then let $E'_t(b) \supseteq E'_t$ be a maximal matching in $E_t(b)$, for $b = 0,1$. Next, we choose labeled edges. Edge $\{i,j\}$ is put into $E_{0,1}$ with label t if there is some $b \in \{0,1\}$ such that in C_b this edge is used for exchanging messages $m_{i,t}(b)$ and $m_{j,t}(b)$ but in the other computation $C_{\bar{b}}$ this edge either is not used or different messages flow across it. We need the following observations.

Claim 1. If $i \neq i_0$ and $j \neq i_0$, and $\{i,j\}$ has label t, then there is some $l \in V$ such that either $\{l,i\}$ or $\{l,j\}$ is in $E_{0,1}$ with a label $t' < t$.

Claim 2. If $i \neq i_0$, then node i is incident with some labeled edge.

The proofs of these claims are similar as in the 1-way case. Now, we eliminate some labeled edges as follows. For each node $i \neq i_0$, choose the unique incident labeled edge with the smallest label, and direct this edge towards i. The resulting set of $n-1$ directed edges is called $E^*_{0,1}$. Using Claim 1, is easily seen that if $(i,j) \in E^*_{0,1}$ with label t, then either $i = i_0$ or there is an edge $(l,i) \in E^*_{0,1}$ with label $t' < t$. This implies that $E^*_{0,1}$ forms a directed spanning tree of G with root i_0. Again, by the definition of $E_{0,1}$, at most two directed edges with the same time stamp can leave any node. This means that from $E^*_{0,1}$ we obtain a 2-broadcast protocol by defining $E_t = \{(i,j) \mid (i,j) \in E^*_{0,1}$ and (i,j) has label $t\}$. □

We leave it to the reader to work out the details of the following argument. We wish to apply Theorem 5.8 to PRAMs with Arbitrary Read and Arbitrary Write with

acknowledgement. To this end, we model both processors and cells as processors in a network. The aim is to inform all p processors of one bit initially located in one cell. If we carry out the construction of the edge set $E^*_{0,1}$ as in the previous proof, it turns out that the resulting tree splits into levels, which alternatingly correspond to PRAM processors only and to PRAM cells only. Due to the special properties of the read operation there can be at most one edge with time stamp t running from a "cell node" to a "processor node" at a higher-numbered level, whereas from a "processor node" two edges with the same time stamp may emanate. This implies that the recurrence inequalities mentioned at the beginning of this section also hold here. In this way we obtain the following generalization of the main result from [3]. (A similar result for the related ERCW PRAM model has been stated in [24].)

Theorem 5.9. *Broadcasting a bit from one cell to all p processors in a PRAM with the* Arbitrary Read *rule and the* Arbitrary Write *rule with acknowledgement takes* $\log_{2+\sqrt{3}} p \pm O(1)$ *steps. The same bound holds for broadcasting a bit on a DMM with p processors and p memory modules.* □

Theorem 5.10. *Any randomized algorithm for broadcasting a bit from one cell to all p processors in a PRAM with the* Arbitrary Read *rule and the* Arbitrary Write *rule with acknowledgement takes* $\Omega(\log p)$ *steps. The same bound holds for broadcasting a bit on a DMM with p processors and p memory modules.*

Proof (Sketch). From a randomized algorithm that performs broadcasting of one bit, such that with probability $1-\varepsilon$ after T steps all processors know the result, one obtains, by standard methods, a deterministic algorithm that performs broadcasting a bit to $(1-2\varepsilon)p$ processors in T steps. To this algorithm we can apply the method for proving the previous theorem. □

Acknowledgements

The author wishes to thank Danny Krizanc for his thorough reading of the manuscript and many helpful and critical remarks. Moreover, the very careful reading and the thoughtful remarks by the anonymous referees are gratefully acknowledged.

References

[1] A. Bagchi, E.F. Schmeichel, S.L. Hakimi, Parallel information dissemination by packets, SIAM J. Comput. 23 (1993) 355–372.
[2] A. Bar-Noy, S. Kipnis, B. Schieber, An optimal algorithm for computing census functions in message-passing systems, Parallel Process. Lett. 3 (1993) 19–23.
[3] P. Beame, M. Kutyłowski, M. Kik, Information broadcasting by exclusive–read PRAMs, Parallel Process. Lett. 1 & 2 (1994) 159–169.
[4] G. Belting, Untere Schranken für die Berechnung von Booleschen Funktionen in vollständigen Prozessornetzwerken im Telefon- und Telegraf-Modus, Diplomarbeit, Universität–Gesamthochschule–Paderborn, Paderborn, 1994.
[5] J. Bruck, C.-T. Ho, Efficient global combine operations in multi-port message-passing systems, Parallel Process. Lett. 3 (1993) 335–346.

[6] S. Bublitz, U. Schürfeld, I. Wegener, B. Voigt, Properties of complexity measures for PRAMs and WRAMs, Theoret. Comput. Science 48 (1986) 53–73.
[7] S. Cook, C. Dwork, R. Reischuk, Upper and lower time bounds for parallel random access machines without simultaneous writes, SIAM J. Comput. 15 (1986) 87–97.
[8] M. Dietzfelbinger, M. Kutyłowski, R. Reischuk, Exact lower bounds for computing Boolean functions on CREW PRAMs, J. Comput. System Sci. 48 (1994) 1231–1254.
[9] M. Dietzfelbinger, M. Kutyłowski, R. Reischuk, Feasible time-optimal algorithms for Boolean functions on exclusive-write parallel random-access machines, SIAM J. Comput. 25 (1996) 1196–1230.
[10] M. Dietzfelbinger, F. Meyer auf der Heide, Simple, efficient shared memory simulations, in: Proceedings of the Fifth ACM Symposium on Parallel Algorithms and Architectures, Velen, Germany, 1993, pp. 110–118.
[11] S. Even, B. Monien, On the number of rounds necessary to disseminate information, in: Proceedings of the ACM Symposium on Parallel Algorithms and Architectures, Santa Fe, New Mexico, USA, 1989, pp. 318–327.
[12] F. Fich, The complexity of computation on the parallel random access machine, in: J.H. Reif (Ed.), Synthesis of Parallel Computation, Morgan Kaufmann, San Mateo, 1994, pp. 843–899.
[13] P. Fraigniaud, E. Lazard, Methods and problems of communication in usual networks, Discrete Appl. Math. 53 (1994) 79–134.
[14] L.A. Goldberg, M. Jerrum, T. Leighton, S. Rao, Doubly logarithmic communication algorithms for optical-communication parallel computers, SIAM J. Comput. 26 (1997) 1100–1119.
[15] L.A. Goldberg, Y. Matias, S. Rao, An optical simulation of shared memory, SIAM J. Comput. 28 (1999) 1829–1847.
[16] S.M. Hedetniemi, S.T. Hedetniemi, A.L. Liestman, A survey of gossiping and broadcasting in communication networks, Networks 18 (1986) 319–349.
[17] I. Höltring, Broadcast und Gossip in parallelen Netzwerken, Diplomarbeit, Universität–Gesamthochschule–Paderborn, Paderborn, 1994.
[18] J. Hromkovič, C.-D. Jeschke, B. Monien, Optimal algorithms for dissemination of information in some interconnection networks, Algorithmica 10 (1993) 24–40.
[19] J. Hromkovič, R. Klasing, B. Monien, R. Peine, Dissemination of information in interconnection networks (broadcasting & gossiping), in: D.-Z. Du, D.F. Hsu (Eds.), Combinatorial Network Theory, Kluwer Academic Publishers, Amsterdam, 1996, pp. 125–212.
[20] R.M. Karp, V. Ramachandran, Parallel algorithms for shared-memory machines, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity, Elsevier, Amsterdam, 1990, pp. 869–941.
[21] R. Klasing, B. Monien, R. Peine, E. Stöhr, Broadcasting in butterfly and deBruijn networks, Discrete Appl. Math. 53 (1994) 183–197.
[22] D.W. Krumme, G. Cybenko, K.N. Venkatamaran, Gossiping in minimal time, SIAM J. Comput. 21 (1992) 111–139.
[23] R. Labahn, I. Warnke, Quick gossiping by multi-telegraphs, in: R. Bodendiek, R. Henn (Eds.), Topics in Combinatorics and Graph Theory, Physica-Verlag, Heidelberg, 1990, pp. 451–458.
[24] P.D. MacKenzie, V. Ramachandran, ERCW PRAMs and optical communication, Theoret. Comput. Sci. 196 (1998) 153–180.
[25] F. Meyer auf der Heide, C. Scheideler, V. Stemann, Exploiting storage redundancy to speed up randomized shared memory simulations, Theoret. Comput. Sci. 162 (1996) 245–281.
[26] N. Nisan, CREW PRAMs and decision trees, SIAM J. Comput. 20 (1991) 999–1007.
[27] D. Richards, A.L. Liestman, Generalizations of broadcasting and gossiping, Networks 18 (1988) 125–138.
[28] V.S. Sunderam, P. Winkler, Fast information sharing in a complete network, Discrete Appl. Math. 42 (1991) 75–86.
[29] G. Tel, Topics in distributed algorithms, Cambridge International Series on Parallel Computation, Vol. 1, Cambridge University Press, Cambridge, 1991.
[30] G. Tel, Introduction to Distributed Algorithms, Cambridge University Press, Cambridge, 1994.
[31] U. Vishkin, A. Wigderson, Trade-offs between depth and width in parallel computation, SIAM J. Comput. 14 (1985) 303–314.